# CODING HORROR

programming and human factors

Custom Search



## RESOURCES



26 Feb 2007

# Why Can't Programmers.. Program?

I was incredulous when I read this observation from Reginald Braithwaite:

> Like me, the author is having trouble with the fact that 199 out of 200 applicants for every programming job can't write code at all. I repeat: *they can't write any code whatsoever*.

The author he's referring to is Imran, who is evidently turning away lots of programmers who can't write a simple program:

> After a fair bit of trial and error I've discovered that people who struggle to code don't just struggle on big problems, or even smallish problems (i.e. write a implementation of a linked list). *They struggle with tiny problems*.
>
> So I set out to develop questions that can identify this kind of developer and came up with a class of questions I call "FizzBuzz Questions" named after a game children often play (or are made to play) in schools in the UK. An example of a Fizz-Buzz question is the following:

> Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".
>
> Most good programmers should be able to write out on paper a program which does this in a under a couple of minutes. Want to know something scary? **The majority of comp sci graduates can't. I've also seen self-proclaimed senior programmers take more than 10-15 minutes to write a solution.**

Dan Kegel had a similar experience hiring entry-level programmers:

> A surprisingly large fraction of applicants, even those with masters' degrees and PhDs in computer science, fail during interviews when asked to carry out basic programming tasks. For example, I've personally interviewed graduates who can't answer "Write a loop that counts from 1 to 10" or "What's the number after F in hexadecimal?" Less trivially, I've interviewed many candidates who can't use recursion to solve a real problem. These are basic skills; anyone who lacks them probably hasn't done much programming.
>
> Speaking on behalf of software engineers who have to interview prospective new hires, I can safely say that we're tired of talking to candidates who can't program their way out of a paper bag. If you can successfully write a loop that goes from 1 to 10 in every language on your resume, can do simple arithmetic without a calculator, and can use recursion to solve a real problem, you're already ahead of the pack!

Between Reginald, Dan, and Imran, I'm starting to get a little worried. I'm more than willing to cut freshly minted software developers slack at the beginning of their career. Everybody has to start somewhere. But **I am disturbed and appalled that any so-called programmer would apply for a job without being able to write the simplest of programs.** That's a slap in the face to anyone who writes software for a living.

The vast divide between those who can program and those who cannot program is well known. I assumed anyone applying for a job as a programmer had already crossed this chasm. Apparently this is not a reasonable assumption to make. Apparently, FizzBuzz style screening is *required* to keep interviewers from wasting their time interviewing programmers who can't program.

Lest you think the FizzBuzz test is too easy – and it is blindingly, intentionally easy – a commenter to Imran's post notes its efficacy:

> I'd hate interviewers to dismiss [the FizzBuzz] test as being too easy - in my experience it is genuinely astonishing how many candidates are incapable of the simplest programming tasks.

**Maybe it's foolish to begin interviewing a programmer without looking at their code first.** At Vertigo, we require a code sample before we even proceed to the phone interview stage. And our on-site interview includes a small coding exercise. Nothing difficult, mind you, just a basic exercise to go through the motions of building a small application in an hour or so. Although there have been one or two notable flame-outs, for the most part, this strategy has worked well for us. It

lets us focus on actual software engineering in the interview without resorting to tedious puzzle questions.

It's a shame you have to do so much pre-screening to **have the luxury of interviewing programmers who can actually *program***. It'd be funny if it wasn't so damn depressing. I'm no fan of certification, but it does make me wonder if Steve McConnell was on to something with all his talk of creating a true profession of software engineering.

Due to high volume, comments for this entry are now closed.

NEXT
**FizzBuzz: the Programmer's Stairway to Heaven**

PREVIOUS
**You Want a 10,000 RPM Boot Drive**

## Written by Jeff Atwood

*Indoor enthusiast. Co-founder of Stack Overflow and Discourse. Disclaimer: I have no idea what I'm talking about. Find me here: http://twitter.com/codinghorror*

Continue Discussion 297 replies

**Toepopper** Feb '07

> **Notice from @riking**:
>
> Please put all solutions to FizzBuzz –here– instead of in this topic. Alternately, don't do it at all - we've clearly had enough proof that it's possible.
>
> (Existing comments from before Discourse containing only

> solutions were moved there.)

*The original post by @Toepopper follows. *

---

Very very common, alas. I once interviewed a candidate for a VBA job (yes, you can stop booing for the peanut gallery) whom I asked to swap two variable contents without using a temp variable. It's the standard a=a+b, b=a-b, a=a-b problem. His answer? Well, I can't do it in VBA, but if you let me use Excel I can put the values in two cells and swap the cells' contents using a third cell.

We hired the guy who said, well, "if they're integers, then I'd do it by a=a|b, b=a^b, a=a^b. But I don't know how to do it if they're strings."

3 replies

---

**AndyToo** Feb '07

I've been making a decent living from programming for over ten years, and if I may say so, I write some damn good code.

However, I have never *once* used - or had call to use - recursion to solve a problem, since I learned about it at university. Does this make me a bad programmer? Or is it simply that people program in different ways for different problem spheres?

I simply can't believe that the other 199 people tested who can't write a simple for…loop have no career as some kind of programmer ahead of them.
Does that mean that every one of those people who do program but don't use (say) recursion is a bad programmer?

Or is it that you can't rely on any one testing method (like writing simple programs) to prove either way that the person can program effectivley or not?

The most obvious way to decide - for me - is to run through these tests, emply the person you like the best and then look at the code they've produced after a week. Then you'll know if they can do what you need.

The bottom line is, you can always learn to pass interview tests of any kind - that doesn't mean you can program or not.

1 reply

---

**James** Feb '07

I think the use of recursion probably depends on the particular field that you code in.

As a web developer I also have very little cause to use it. The

only time I use it is when I need to search through the file system.

**VibhavS** Feb '07

Isn't the code for that …

int main() { ExitPaperBag(); return 0; }

I'm not sure … copied it from somewhere 😛

**Moz** Feb '07

There's a cynical voice in my head that says "because no-one has ever asked them to".

Where I work we spend a long time looking before we hire someone, because it's so hard to find them. We use a walk-through test that has a sequence of "questions" leading to a simple app and are willing to argue the toss (I was hired despite arguing that the test had significant flaws: "why would I do that? That's stupid because…" when they wanted me to code a solution that led to the next question).

I fear that any take-home or pre-interview component would lead to plagarism or "helped" solutions, so we make candidates perform in front of us.

That's got our ratio up to 3/4 good hires.

1 reply

**James** Feb '07

shenpen - Unfortunately that wouldn't be a valid solution (in VBScript) because WScript.echo adds a CRLF. In another language it might work though.

**MichaelR** Feb '07

Sorry, but if someone were to ask me how to swap two variables w/o a temp variable I'd ask them to give me a good reason why.

Not being able to answer that particular question certainly doesn't preclude someone from being a good programmer. That'd be like asking a C# programmer how to do modulo 16 using only a logical and. Why the hell would they need to know that, and how does that help you determine that they understand the ASP.Net framework, etc.?

**Mike_Miller** Feb '07

shenpen: As you point out, your solution doesn't handle line breaks. It also doesn't print out the original number!

**MikeW** Feb '07

shenpen, for shame. Read the spec again.

I don't think I've ever interviewed a programmer quite that bad, but I did have a conversation with a contractor I was replacing once: I'd introduced a class into the monster spreadsheet he'd been maintaining and he looked at it and remarked "I wish I knew how to do that". He was picking up about GBP400 a day (call it about $650 at that time) for this.

As I'm about to start looking for a programmer I shall be able to implement my long-cherished plan of asking candidates to submit a page or so of what they consider to be "good code". I don't mind if they wrote it or not, I want to see if we agree on the look of "goodness". If we get that far, I can find out if we agree on why it's good.

This is nasty but it's technically a one-liner:

1.upto(100) { |n| puts n % 3 == 0 ? n % 5 == 0 ? "fizzbuzz" : "buzz" : n % 5 == 0 ? "fizz" : n }

**Gareth** Feb '07

WOW! (and not the vista kind).

I can't believe there are people out there applying for jobs in software that cannot write a fizzbuzz program.

**Mike_Miller** Feb '07

It's true that knowing how to do modulo 16 using AND doesn't relate to understanding ASP.Net.

On the other hand, do you want an ASP.Net technician, or someone who understands the theory behind it? Can you really call the candidate who only knows ASP.Net, but nothing about a basic logic a programmer (or a computer scientist, or a software engineer, or a coder), or is he just some guy who knows ASP.Net?

**codinghorror** Feb '07

James: it's amusing to me that any reference to a programming problem-- in this case, FizzBuzz-- immediately prompts developers to feverishly begin posting solutions.

1. The whole point of the article is about WHY we have to

ask people to write FizzBuzz. The mechanical part of writing and solving FizzBuzz is irrelevant.

2. Any programmer who cares enough to read programming blogs is already far beyond such a simple problem. It's the ones we can't rerach-- the programmers who don't read anything-- that we have to give the FizzBuzz test to.

But it's OK, I mean no offense. It's just funny. I know how software developers' minds work. 🙂

**MichaelR** Feb '07

My point is that what you're asking for isn't "domain knowledge" strictly speaking. Using an XOR to swap two variables w/o a temp variable is only really useful when memory is expensive (embedded programming, et al). If you're not interviewing for an embedded programmer why are you worried about it?

Same with the modulo trick. The first time I ever encountered that I was writing a Perlin Noise Generator based off of someone else's work. Outside of routines that need to be highly optimized there's no reason for it.

Are you trying to hire an ASP.Net programmer or an embedded systems programmer? Granted, the XOR trick is pretty well known, but not knowing it off the top of your head says nothing about your abilities as a programmer (whereas the fizzbuzz example does).

No one can know everything. I would expect that type of attitude from a non-IT person, but not someone who is interviewing programmers.

**James** Feb '07

Ha ha! I see what you're saying Jeff.

I hope I didn't come across as feverish though. I sincerely thought that a solution may be of interest to someone who couldn't write one themselves.

Perhaps you have readers who aspire to be programmers but are still learning?

Perhaps a future Google search for "FizzBuzz" will bring back this page? (And the Googler might be after the solution).

Interestingly, the use of a technical test in interviews can actually work both ways. In my recent hunt for a new job I automatically turned down any vacancy where I was *not* given

a technical test. My reasoning behind that is that I know *I* can program, but do I want to work with *other* programmers who haven't been tested?

The harder the test I was given, the more excited I was about the vacancy.

If anyone out there is looking for a new job I urge you to bear that in mind.

**LKM** Feb '07

Instead of using recursion, it's often faster to use a Stack.

**Grant** Feb '07

There's always 'Survivor for Developers' for those who don't make the cut.

> `http://techtalkblogs.com/blog/archive/2007/01/26/1837.aspx`
>
> I once heard of a development group where every three months a vote was taken and the bottom ten percent of the group were replaced by newcomers. It sounded a bit like Survivor - Development.
>
> A common immediate reaction is to assume that it would become a popularity contest but that wasn't the case. The group had very specific KPI's to achieve and those that helped achieve those goals were well respected in the group, regardless of "popularity". Really anti-social folk tended to get removed though. (In our company, one of the tests we use during the hiring phase is lovingly called the "no axe murderers" test. I think we have Chris Hewitt to thank for that awesome terminology.)
>
> When working as a manager, I always felt that Australian laws made it very difficult to remove the dead-wood out of teams. Clearly, having a tribal council and an eviction for certain development groups wouldn't work, but on this Australia Day public holiday, ask yourself this: If your team had a vote yesterday, would you still have a job on Monday? If not, do something about that now.

At the end of each week, you vote the worst dev off the island.

**obivous** Feb '07

It's just performance anxiety.
Most people are terrified/stressed out during interviews.
Not "Can you X?" -

“Can you X whilst terrified?”

**rien** Feb '07

AndyToo wrote:
“However, I have never *once* used - or had call to use - recursion to solve a problem, since I learned about it at university. Does this make me a bad programmer? Or is it simply that people program in different ways for different problem spheres?”

i think you missed the point: if ou were explicitly tasked to write a algorithm using recursion, would you be able to do it ? would you be able to recognise the use of recursion while reading someone else code ? and most important, the one i struggled teaching to newly hired guys and which most people dont learn at school: do you understand the principles behind recursion ?

every good programmer knows there are plenty of ways to avoid recursion, but being able to avoid it implies that you know what recursion is.

and that is the point of such a question during an interview.

**Kristoffer** Feb '07

Damnd thing took me three minutes to write in php. I am getting slower with old age 😉

if ($x/3 == floor($x/3) … and so on.

**Tonetheman** Feb '07

I wonder why it is 199 out of 200 instead of more reasonable number like 99 out of 100? In any case I believe him. It is amazing how many people are paid to be programmers that struggle at the job and clearly should have been hired in the first place. When I used to teach college I noticed the same trend… a large number of CS student just really did not understand how to program. It was scary.

**t1110** Feb '07

The majority of comp sci graduates can’t. I’ve also seen self-proclaimed senior programmers take more than 10-15 minutes to write a solution

Ofcourse we all know that being a elite programmer is all about how fast you can program, especially in a high-pressure situation.

**HaX80r** Feb '07

Wow. Now that's sad. I don't call myself a programmer at all, and I could write a solution to this in Qbasic!

**Me137** Feb '07

Most development is simple viewing and editing of data in forms.

Developers who can't write "fizzbuzz" can be productive in such an environment, especially if there is existing code they can copy and modify.

**Me138** Feb '07

Most development is simple viewing and editing of data in forms.

Developers who can't write "fizzbuzz" can be productive in such an environment, especially if there is existing code they can copy and modify.

I suspect some of the people above complaining about the high pressure of having to write a ten line program including division may fall into this category. 🙂

**shiva** Feb '07

if program does require recursion, then definitely everyone will use this.

**Malixu** Feb '07

"Most development is simple viewing and editing of data in forms.

Developers who can't write "fizzbuzz" can be productive in such an environment, especially if there is existing code they can copy and modify."

They'd also be an incredible liability. If fizzbuzz is beyond them, concepts such as memory/processor usage, security, stability, defensive programming, etc. are going to way out of their grasp. I've worked with someone like this; they *cost* us time, because *everything* they wrote had to be completely re-written by someone else. Unchecked inputs, error messages that consisted solely of dancing cats (seriously), SQL and HTML injection holes all over the place.

On writing a program to switch two variables without a temporary variable; while I had a hunch it involved bit operations of some kind, I certainly wouldn't know how to do it

operations of some kind, I certainly wouldn't know how to do it off hand.

On time taken to program; I'd probably spend several minutes just staring the FizzBuzz task, looking for some complexity I'd missed, so that's also something to take into account.

**rien** Feb '07

to LKM:
"Instead of using recursion, it's often faster to use a Stack"
do you understand how, in almost any language, recursion involves a stack ? you are just making the hidden part visible…

to shiva:
"if program does require recursion, then definitely everyone will use this."
agreed, but you are free to think a bit longer and see if you cannot avoid it. this is the difference between a good programmer and a smart programmer: the former knows how to solve the problem while the later invents new ways to solve the problem more efficiently.

to Fabian:
no, you forgot a part of the requirements. wroooong ! do it again !

to Jeff:
"it's amusing to me that any reference to a programming problem-- in this case, FizzBuzz-- immediately prompts developers to feverishly begin posting solutions."
it is also amusing how many of those solutions are wrong. so typical of our profession…

**Chris** Feb '07

Back in '99, I was given a FizzBuzz task by a Microsoft recruiter to design a program that would take two inputs and determine if they were anagrams of each other. I began with a quick test of string length - if they didn't have the same number of characters then it was immediately rejected. He was very impressed that I would start with such a test. Seems most people would go through cycles and cycles regardless of the inputs.

**Omar** Feb '07

To be honest as a recent grad and somone who is new to being a software developer I think if you are incompetant you shouldn't last long anyway.

In my first month at my new job I had to learn VB and C++ from scratch, two months later I was moved on to a project in

from scratch, two months later I was moved on to a project in C# using ASP.Net which I'd never used before. I've seen similar tests to the fizzbang questions, and they are useful.

My current employer gave me 3 hours to parse an XMl file and display it as a tree structure. I'd not spent much time playing with XML before but with after a quick google for a reference on the SAX api I was coding away. apparently half the candidates couldn't write anything and others didn't know how to use and IDE and insisted on using a text editor. which they screwed up.

Another couple of employers has written tests with small "write a code fragment to do this" and "spot the mistake in this code" questions. both were piss easy but there were some trick questions in there.

I think what should be tested especially for recent grads is not the knowledge per say, but the capacity to pick stuff up, I got my current Job becuase I took an API I had never used before and figured out how to use it. I'm by no means a pro, but I have the capacity to learn by myself, some people can't pick up new things unless they are spoon fed.

University is not an excercise in cramming your head full of knowledge it's an excercise in learning how to learn quickly and efficiently.

When you know the basics you can teach yourself a new language in a few hours, and depending on the complexity learn a new API in an hour to day. Many grads I find dont have that capacity, I did a masters with a bunch of people who came from different universities to me, and some of them just couldn't cope with having to fend for them selves.

Thats my two cents anyway. I think the fizbang questions help sift out the muck but the problem is more intrinsic and needs to be addressed at the level of academic institutions.

**Ade** Feb '07

Stu Thompson wrote, "There is a stunningly large population of people who describe themselves as 'professional software developers' who have not the faintest idea about recursion, bit masking, hexadecimal maths…or how to code a simple for loop that does something like FizzBuzz."

FizzBuzz, sure, any programmer should know how to write that. Bit masking though? That example just goes to show the wide range of expertise involved in programming, not all of which is required to be a successful or skilled programmer.

I don't think I've used bit masking (and I can't even remember what it is, exactly) since college. But Stu goes on to say talk

about asking for SQL examples in interviews, and here not only would I be able to demonstrate inserts and updates - of course - but left joins, cross joins, multi-table deletes, whatever.

FizzBuzz is an acceptable test because it is non-domain specific, the same cannot be said for all other types of programming knowledge.

**Mike_Miller** Feb '07

Quote: Granted, the XOR trick is pretty well known, but not knowing it off the top of your head says nothing about your abilities as a programmer (whereas the fizzbuzz example does).

True. However, someone who's never read enough professional literature (including programming blogs) to have seen this may either be:
a. very inexperienced OR
b. not interested in programming outside of coursework/their job.

I submit that such a person is never going to be a great programmer (although they may be, barely, adequate).

**coderprof** Feb '07

Common, unfortunately. The majority of CS grads can handle the trivial stuff like "FizzBuzz" (though a slight majority probably never really grokked recursion). But I've seen people hired at my last job who couldn't code.

in one case, the person did know a bit of syntax, but they had absolutely no debugging skills. This person didn't know the integrated debugger existed. They didn't even use simple debugging outputs to the console. No, their method of debugging was to randomly change stuff to see if it fixed their problem. Not surprisingly, that didn't work very well.

**Tatsky** Feb '07

Fabian

Nice little snippet there. However, it doesnt fulfill the specification.

"Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz"."

You arent writing out i when its not %3 and/or %5

Close but no cigar.

**natron** Feb '07

I need a raise then, cause I actually write and ship a COTS version of the FizzBuzz application.



**CuRoi** Feb '07

"it is also amusing how many of those solutions are wrong. so typical of our profession…:

I noticed this as well. Makes one really start to question the worth of such statements as “if you’re reading this blog, you’re already much better than those *other* programmers”.

**MartinP** Feb '07

I remember at my first University course the teacher asking how many people never programmed before. Many hands were raised, and I felt bad for them for choosing a future profession without knowning what it actually was about. Then he asked how many people never touched a keyboard before. How shock was I to see at least 10 persons raise their hand, mostly women.

**DavidA** Feb '07

I’ve witnessed the same, sad truth. It even inspired me to write my first blog entry on candidates attempts to code strcpy(): http://davidavraamides.net/blog/2005/05/07/strcpy/

I think the takeaway from this is that you are a fool if you are interviewing programmers and not requiring them to write some code.

Toepopper: swapping variables without a temp is more what I’d call tech trivia then useful knowledge. There really isn’t a good use case for it anymore (its not even necessarily faster than using a temp since procs have so many registers these days). Besides, your solution of a=a+b, b=a-b, a=a-b fails in overflow cases. The candidate’s XOR solution is more correct, but still unnecessary.

rien: you are right on regarding *understanding* recursion. I agree that its frequently not necessary and often the less efficient solution, but being able to recognize that fact, and to read someone else’s code, requires familiarity with the technique. And that’s why I would consider it fair game in an interview.

**Dennis68** Feb '07

“we all know that being a elite programmer is all about how fast you can program, especially in a high-pressure situation.” … well, yes, if you’ve got an online application and somebody’s just discovered a serious bug, sometimes that’s exactly what it’s about.

Recursion? I never used to use it either. Then I read the first chapter of Structure and Interpretation of Computer Programs, and some other stuff about writing compilers and interpreters. Now I use recursion all the time, coming up with all sorts of general, automated solutions for the sorts of tedious tasks I used to code out by hand.

**John_Pirie** Feb '07

Why can’t they program? Because demand for good programmers has outstripped supply. Those of us who interview a lot of applicants (I’ve seen at least 50 the last year) know two things: most can’t program, and those who can will be tough to get. And now it’s been bad enough for long enough that many doing the hiring can’t program either, so screening is often incompetent and these charlatans stand an excellent chance of getting hired anyway.

Testing is imperative, but some just seem reluctant to do it – it seems so cold and mean.

Fizzbuzz looks like a good test. Here are two other standard questions, the first one broadly published and the second very general. Many “programmers” flop on one or both:

1. Draw four “cards”, showing “A”, “B”, “2” and “3” respectively. Say that for each card there is a letter on one side and a number on the other side. I have a theory that if there is a vowel on one side of a card, then there will be an even number on the other side. Which cards do we need to turn over and examine the other side of to adequately test my theory?

2. What is a hash table, and when would you use one? Why might you prefer using a hash to using an unsorted array, or a sorted array? What tradeoffs are you making when you choose between a hash and a sorted array?

When the applicant doesn’t know what a vowel is, or a hash table is, I know it’s going to be a long day…

**David** Feb '07

Hrmf!

I wouldn't hire any of you guys who sent in the Fizz/Bang program. You should have declared 5 and 3 constants, and used the constants in your code.

Besides, real programmers wouldn't program in VB, they'd use Perl. Now, if I can only get this page to work in my Lynx web browser.

**Malixu** Feb '07

"…well, yes, if you've got an online application and somebody's just discovered a serious bug, sometimes that's exactly what it's about."

Ugh, had one of those on Friday. Our caching system was spitting out the wrong data (by which I mean, the wrong user's data). Never showed up on the dev system because the caching system uses SoftReferences to allow the Java garbage collector to expire data from it, and the dev version never ran long enough for garbage collection to really be an issue. Important lesson about being *really* sure about your compareTo() methods before using java.util.TreeMap.

Took us 40 minutes to locate, fix, and patch to live. Which is a little longer than usual, but locating the problem more or less came down to eliminating everything it couldn't be, then figuring out how it was the fault of what was left.

**kg2v** Feb '07

Geez guys - EVERY ONE of you who gave example code - EVERY ONE - hard coded the FIZZ and BUZZ conditions… And a LOT of you wrote code that will do the FIZZ, BUZZ and FIZZBUZZ but NOT print the integer…

**Vizeroth** Feb '07

My biggest problem with solving simple programming problems tends to be the memorization involved with different languages, especially if I haven't used a particular language in a while. More often than not I'll go through my resume and remove (or limit mention of) languages that I used for only a short time or simply haven't used in a while.

For example, I feel quite competant with dealing with basic database issues, but without having touched a database in a few years I'd have to look up even the most basic sql statements, so I wouldn't claim to be a capable database programmer, though after a couple of weeks I could be at least a moderately good one.

Recursion is a similar problem with different languages. Some languages don't allow it at all, and others have very specific restrictions that make it a little harder to just jump in and say we'll just make this call here and make sure the conditions are in line to stop the whole mess from self-destructing.

In other words, I have a terrible memory, but I can still answer most of those questions with whatever language I've been using recently. I simply have to maintain my resume to avoid being asked questions that would send me to the books (or the web) to find an answer that should just spring to mind immediately.

**Danimal** Feb '07

"And did you really think you could say "only 0.5% of programmers can write FizzBuzz" without people showing you how they do it? :P"

Gareth: my guess is the ones who couldn't do it didn't bother to answer

**l0b0** Feb '07

Anecdotal "evidence" against this: When I started my first job after university (also, my first programming job), I could /maybe/ have answered FizzBuzz in Java. I'd programmed probably less than 1000 lines my whole life. Two months later I was improving my colleagues' code in a language I'd never used before. 2.5 years, some 36,000 lines of production code, and a good recommendation letter later, I'm stuck in a job market that only wants people with 5 years experience.

**Eam** Feb '07

"Ofcourse we all know that being a elite programmer is all about how fast you can program, especially in a high-pressure situation."

So you think that, even in an interview, it's alright for an experienced professional to take 10-15 minutes to write FizzBuzz? That's more than one minute per line!

**LKM** Feb '07

do you understand how, in almost any language, recursion involves a stack ? you are just making the hidden part visible…

Do you understand that by using a local stack instead of the call stack, the computer needs to keep track of much less information and needs to do much less work building and

tearing down the call stack? In most languages, using a stack instead of recursion speeds up your app tremendously.

Also, why the hostility? Did I somehow insult you? If so, I apologize, but I fail to see what exactly I did to you.

**anon84** Feb '07

Any question can be made difficult. If asked an easy question, an interviewee knows they have to get it right and will become more nervous as time goes on. Over the course of three minutes, it can get *very* bad.

“Write a program that prints the numbers from 1 to 100.”

Inclusive/exclusive? Are we in a language in which fencepost errors are likely?

“But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”.”

On the same line or different lines?

“For numbers which are multiples of both three and five print “FizzBuzz”.”

What about performance? The interviewee may be trying to remember how mod is implemented. And thinking about fence posts. And thinking about 3 cases of control flow. And “fizzbuzz”. And working quickly. And acting calm. And why they’re being asked this stupid question. Normally you do not deal with all of these things at the same time, but asking a simple question in a stressful environment in which the interviewee is getting ready to put 110% in to whatever you say will likely lead them in to trying to do so. There’s only so much you can keep in your head, and even less when you’re nervous. As you get more nervous (because the question is taking a long time), your nervousness will feed into itself.

Whenever I’m asked a question like that, I think the manager is clueless, both technically and in their ability to lead a team. Asking insightful questions about harder projects will reveal who did the work, and asking questions appropriate to the applicant will show you understand your team. This just shows you’re a delusional asshole.

One company I know required applicants to do an almost code homework assignment before applying. A code sample or two as well as discussions of past projects should suffice and probably bring up good topics, as the subject will reveal what they’re interested in. If you’re a coder on the same project, ask for their solutions to problems you’re having.

I stopped explicitly listing my technical skills recently and just

put projects I have worked on - that also cut back on the bullshit questions people asked. When not given something distracting like that, interviewers have to think about better topics - and the interview may even become a fun conversation at that point.

**fabien1** Feb '07

It requires a huge deal of self-confidence to be as much effective during an interview as during your lazy sundays. Most of us will think twice before considering writing the *simplest* statement in front of an interview panel. I agree with most of this article - but it should really insist of the psychological factors (Of course, you could argue whether good programming abilities are dissociable or not from good social and communication abilities).

**mrprogguy** Feb '07

Mike Reiland: when you wrote

“Not being able to answer that particular question certainly doesn’t preclude someone from being a good programmer. That’d be like asking a C# programmer how to do modulo 16 using only a logical and. Why the hell would they need to know that, and how does that help you determine that they understand the ASP.Net framework, etc.?”

I think you missed the bloody point. There are plenty of people who “know the [fill in the blank] framework,” but can’t program worth a flying damn in a circus tent. I’ve worked with people who brought a Master’s Degree in Computer Science to the table, but couldn’t understand the concept of a state machine or even building flag values by accumulating bits. I showed a simple event-driven framework to one junior programmer and his response was “I wasn’t taught that method.” He didn’t ever appear to grasp that there was more to the world than was taught in his Computer Science classes, and that we were allowed to actualy come up with and express ideas all on our own without a professor’s approval. Needless to say, he didn’t last long. (Actually, he was a little on the weird side; he kept coming to work after we fired him. That concept of being fired seemed beyond him as well.)

Check out http://worsethanfailure.com/Default.aspx for examples of people who “know the [f.i.t.b.] framework,” but can’t seem to express the simplest of concepts in code. It has little to do with hoop-jumping and clever tricks, and everything to do with getting the job done on a daily basis: some of these people come out of University with a degree and no idea at all how to code.

The company where I work has an extensive series of programming tests for potential new hires, including specific problems derived from the kind of coding we expect them to do on a daily basis. It's amazing how many people can't get through it. One applicant excused himself to go to the bathroom during the middle of the test, and never came back. It's not even a hard test.

For the record: wrote the FizzBuzz thing in under two minutes– right the first time. (In all fairness, though, I first started programming when I was 15, and that was 31 years ago. If I can't do FizzBuzz after that amount of time, I need to hang up the cape.)

**EdwardM** Feb '07

I think it's interesting to see how many people here are innately drawn to developing a solution to the problem, completely overlooking the original point (which was that many candidates wouldn't normally find themselves innately drawn to this).

I think this defines one key aspect of programmers who succeed, and those who simply maintain as they move forward: an inherent interest in/drive toward practicing (in the Dave Thomas "Code Kata" sense) their craft, vs. someone who simply views programming as a job to complete.

In any field, there are those who are passionate about what they do, and there are those who do it because they need something to do; it's just starting to catch up en masse with technology fields now. And it's not necessarily a bad thing: you always need people who would rather do repetitive tasks, and you always need people who would rather be tackling ambiguously-defined problems, and these people are rarely one in the same.

And, just because I'm a geek, and can't help myself, here's what I whipped up in a couple of minutes. I'm sure it could be better, but that's when you start up an interesting conversation about refactoring in the interview, right? 😉

```
for i in range(1, 101):
x = ""
if i % 3 == 0:
x = x + "Fizz"
if i % 5 == 0:
x = x + "Buzz"
if x == “”:
x = str(i)
print x
```

**Kyralessa** Feb '07

Anyone who expects a VBA programmer to write “swap two variables without using a temp variable” code is going to end up hiring a programmer who gets bored in three months and leaves.

The FizzBuzz problem is a good example of taking a simple set of requirements and translating them into program code.

Unless you’re interviewing for an embedded software position, the “swap two variables with no temp” problem is a good example of utterly pointless “Let’s see if you know the same nifty trick I know” interviewing. You might as well ask the interviewee where they bury the survivors when a plane crashes on the border of Canada and the U.S.

**EdwardM** Feb '07

Okay, so python’s required indenting and blogging don’t seem to go well together. Ah well, you all know what I meant. 😉

**tutash** Feb '07

I use recursion every day. It makes handling XML simple. I can’t imagine not using it. However, if I were hiring someone, I’d not make it a prerequisite. Neither would I fail them if they didn’t know what a modulo was. I’d be happy if they could solve FizzBuzz in 30 minutes. I’m more concerned that they understand a little about abstraction. If they don’t know what “this” is, then we’re screwed.

**programmer3** Feb '07

Opps… I suppose i should have a “return(0);” before that last curly bracket…

I guess that’s what I get for compiling without -Wall …

**LKM** Feb '07

I think your man was just pointing out
that it’s a bit dodgy to say “use a
stack instead of recursion”.

Well, that’s not what I said. I said “Instead of using recursion, it’s often faster to use a Stack.” Which is true.

Recursion pretty much always uses a stack,

Yeah, but that does not change what I wrote. Coincidentially, I have written a Java-to-native compiler and happen to know how call stacks are implemented, and the overhead involved in

building them up and tearing them down 🙂

**Ben** Feb '07

A simpler question, requiring NO answer: If you were writing a disk based index for fast access to usernames from id's, what datastructure would you use?

Blank Stare = Fail.
Eyes light up with passion = Hire on the spot.

**Dave** Feb '07

Note that the FizzBuzz Test requires at least some simple number theory thinking. Yes, some small children understand it, but most people with standard schooling are not comfortable with the ideas of multiples, common factors, modular arithmetic, etc.

I only point this out because 199 out of 200 should make us question ourselves about the test. I'll accept that 90% of so-called programmers are "low functioning" but not 99.5%.

Is this FizzBuzz-style number theory knowledge required for programming? No. Does it help? Yes, a lot. I find applied mathematical knowledge to be a better indicator of quality, efficient, expandable programming than a checklist of fad languages and environments.

But, still, watch out for hiring *only* super-nerds or your product will be super clever but unusable by "normal" people. A little bit of variety of backgrounds helps.

**David_H** Feb '07

"Beware of bugs in the above code. I have proven it correct; I have not actually tried it." -Knuth

**Adrian** Feb '07

ha ha. Lets see some ASM code. You got 10 seconds. Assuming 8086.

**James** Feb '07

What a whiny post. You complain about the graduates themselves but what about the departments who taught them? If you were really concerned with the situation, rather then obliquely congratulating yourself on being so shit-hot, why not attack the professors?

PhDs do research, not programming and I've know dozens of

apparently dim graduates turn into excellent developers.

**KeithW** Feb '07

I once interviewed a guy who TAUGHT programming at a city college and HE couldn't answer very basic questions about the language. My typical C++ interview question is to ask whether they understand what happens when an exception is thrown, stack unwinding, how it can cause memory leaks, and what to do about it. (someone out there is thinking the answer is "program in Java" but they should go die now)

**DonS** Feb '07

I suspect the original observation can actually be generalized to almost any other profession/job. It seems a common complaint that one has to search hard for even basic skills and then harder for someone who will put the effort into being an employee that can be counted on to even show up regularly and actually work.
One wonders what the future holds when you read the above posts, realize that a fair number of good programmers will be leaving the market over the next decade (boomers), and hear reports of declining enrollments in the field.

**Steve** Feb '07

I wonder how often this book is read anymore:
http://www.amazon.com/Algorithms-Structures-Prentice-Hall-Automatic-Computation/dp/0130224189

A sad fact is that many of these types of skills often don't apply in business programming, and so you lose touch. Many of the exciting routines we used to learn just don't apply. A little sad.

How about asking people the pros/cons of various types of sorting algorithms?

http://linux.wku.edu/~lamonml/algor/

**PL110** Feb '07

Not sure what kind of educations you have over there but certainly here in Sweden you write A LOT of code if you study computer science on university or college elevel.

Even in high school programming courses you write code from day one, are you seriously saying this is not the case in the US ??

The whole thing sounds very strange to me.

**BrettM** Feb '07

The original post and the original quoted post do nothing except perpetuate the worst stereotype of what a good coder is. Executive managers prefer to think of their best developers as sweaty grubs who pound the keyboard (rapidly, I'm sure) and who amuse themselves with anal debates. Most of the comments here seem to confirm that stereotype willy-nilly.

"I've also seen self-proclaimed senior programmers take more than 10-15 minutes to write a solution."

Perhaps they became senior programmers by acquiring the habit of quiet reflection *before* pounding the keyboard (rapidly). Many executives do not understand that the person staring out the window may be getting more done than the busy people at the keyboard.

Job interviews structured to find sweaty grubs will probably find sweaty grubs…

**Casper** Feb '07

Isn't this an issue because you can learn to people to master a programming language but not how to solve problems? The people who couldn't solve the fizzbuzz test you describe in your article, might be great at solving well defined problems. I remember Jon Udell wrote an article about the tacit dimension of tech support:

http://www.infoworld.com/archives/emailPrint.jsp?R=printThisA=/article/05/06/15/25OPstrategic_1.htmlsource=searchresult

I clearly see "problem solving" as the skill that defines what you expect to be a good programmer.

**Ken** Feb '07

As I'm reading this blog topic and all the responses I'm having a hard time breathing for all the smug in the atmosphere (or, rather, blogosphere.)

Oh, I'm such a great programmer that I can't imagince that a professional programmer can't do hexadecimal maths, says the crowd. Look, I'm a relatively new programmer. I do exclusively VB.Net but I work daily in several applications (some in-house, and some production) and I've been programming now for over a year and a half. It was very hard at first, but I'm becoming more proficient every day. One day, I know, I'll be very good but I'm certainly not there yet. You see, I made a career change. Many of you grew up with computers –

tinkering with them, writing programs when you were eight, etc. I don't have that background. After years in the military I got out and decided I wanted to challenge myself and become a computer programmer. So I went to a two-year college, got a degree and got my foot in the door at a small software company.

If you're asking me, I'll say I've done well. I'm also probably being realistic when I say I'll never work as a hotshot programmer for Microsoft. But, you know, there are plenty of programmers out there who aren't superstars. They don't need to be. They are proficient and reliable and they get the job done. Just not superstars that eat, sleep and breathe programming. Does that mean they don't deserve to be programmers?

I understand the point of this blog – but I just think some of you should step back and take a look at your egos. Because some of you might not have hired me and you'd have lost out on a good programmer. BTW the FizzBuzz thing would have been no problem, but I have no freaking idea what Hexadecimal maths is (it's off to wikipedia!)

Just my two cents. Narf.

---

**lexu1** Feb '07

Jeff, I like your FizzBuzz question, but my use of it in screening would be different, since I see an entirely different problem in hiring a programmer.

I'd try to sit down with the candidate, ask her to do the "FizzBuzz dance" then wait to see what she does next.

I would be disapointed not to hear some questions, like
"why should she write this code."
"should the code be maintainable, fast, memory economic or is it a show piece?"
"might the requirements change."

before she starts coding …

After she presents a first solution, I'd change some requirements, crticise some aspect of the code (is a repeated execution of the modulo function needed/efficient, don't you know more about the sequence of numbers your processing …)… and learn not only "can she code" (in my experience, they usually can…), but can she think, does she stand up for her beliefs, does she spot incomplete requirements, can she point out the faults politely?

Only 10% of the day are taken up by coding … much of the rest of the time is spent interacting with the

designer/architekt/manager/customer… that's where most programmers I've employed in the past have not failed, but struggled.

**Vladekk** Feb '07

Haha, that's why we outsourcers get these jobs instead of some lame US graduates

**PL111** Feb '07

I think this blog entry would have been better off at thedailywtf.com because it just has to be complete BS, if they have any kind of computer course and the paper isn't falsified of course they know how to solve that, if they don't then you have a serious issue in your education system.

Maybe if you stop wasting so much money bombing people and spend it on education instead ?? Just an idea.

**MatthewC** Feb '07

I think the point isn't that any one of us (most likely) could spit out a solution to the Fizzbuzz problem in a minute - I had a solution in my head before I'd really tried to think about it. The point is that people go into 'programming' jobs sometimes for the wrong reasons. Maybe because the money is good. I don't know. But these are exactly the kind of folks that need to be encouraged to look elsewhere - and I mean a different career path - for employment. Encouraging folks with no aptitude for problem solving and logical thinking to work in a field where such an aptitude is needed simply drags down the team.

The problem I've seen in the past is that often those doing the interviews aren't qualified to know the difference between a person in a suit with a smile and a good disposition from a person that will actually be a productive member of an organization. This is particularly the case in larger organizations I've worked within.

Any organization that doesn't actively try to find the best people deserves what they get in my opinion.

**Eric** Feb '07

I can sort of understand this problem, actually (no, not fizzbuzz - that one I get).

I graduated from a major CS school a decade ago - but the school was a Comptuer Science school - heavy on the science - not a computer engineering school. I had classes on Kleene stars and turning machines and all sorts of academic

languages that I was unlikely to see again unless I went into computational linguistics. My token database class spent exactly a week on SQL - and, I suppose, with reason…that's "IT", not "CS" in their view. Of course, since I graduated I've written SQL pretty much every day of my career. We spent a lot of time discussing the finer points of various languages, spent a lot of effort writing applications that used these finer points, but didn't bother to spend much time on "what you would actually use these things for in the Real World." That fell outside the realm of "CS" again. I guess the assumption was that this is academia, not a trade school.

Of course, on the flip side, the IT program at the same school required only the most basic programming classes and a lot of business stuff. CS gave you all sorts of cool architectural techniques but few basic skills to use them, and IT gave you all sorts of practical basics but few techniques. There was a huge range in the middle where "actual programming" fell that was never addressed.

I'm not even going to go into some of the small schools with the build-your-own-degree programs.

The only reason I was ever able to hold a job or write some code immediately out of school was because of some hotshot programmer friends who showed me some stuff. Blind luck, basically.

So collegiate education seems to fall short. But that's only really a small part of the problem. From the day I started my post-college career I've been in the minority in every IT department I've ever been in - I'm usually the only person with a formal background in technology. Many people in your standard corporate IT-development department landed there by accident or by lateral move - their training includes "Teach Yoruself VB4 in 21 Days" or maybe a few classes at the local community college. I've worked with plenty of people who consider copy-paste in visual studio "code reuse", and I wouldn't say this any fault of their own - they've just never been shown what OO or even really decent procedural code looks like.

And there's the web…web development allows a lot of sloppy code to slide under the radar, it seems. If a developer has never had to solve a problem because a web browser front end lets them get away with it, they're never going to think about that problem in the first place.

I'm not sure what the solution really is - it'd be nice to have the education broaden a bit from both directions, but that will only help people who are exposed to it in the first place, which isn't often the case. Working with a good programmer helps immensely - it's what kept me from being useless for years,

and I've seen one guy's good code transform the coding styles of entire companies in a matter of weeks - but how to do you guarantee a good coder in your environment?

Something to ponder, I guess.

**Jim_Rogers** Feb '07

To answer your original question: I've worked on big projects where the low-level programmers (and me, the contractor,) spent weeks or months doing cut, paste, change-the-variable-and-function-names kind of work. A person can sit in front of a computer and "program" for a long time without ever having to actually write even simple logic from scratch.
Maybe this is the "experience" your seeing on people's resumes.

**lall** Feb '07

It is interesting that at least two solutions given in the comments are wrong. People don't even know that they can't program.

**Corey** Feb '07

wow… pretty scary thought.

Interestingly, in the USA (at least in massachusetts where i am from), we used to play the same game but it was called "BizzBuzz", not "FizzFuzz".

So here is an American BizzBuzz version in Python:

**Mike_H3** Feb '07

Can anyone post the answer to the vowel/even number logic puzzle. I hate logic puzzles, but the programmer/pessimist in me says all 4 because you never know what the next input would be. However, I'm guessing the answer he wants is "A" and "2"?

**rien** Feb '07

to LKM:
i am sorry if you felt my message as hostile. english is not my native language and i may sometimes seem a bit rude (i do seem rude anyway on my daily life). it was not meant to be offensive at all, and i deeply apologize if you took it that way.

anyway:
"Do you understand that by using a local stack instead of the call stack, the computer needs to keep track of much less

call stack, the computer needs to keep track of much less information and needs to do much less work building and tearing down the call stack? In most languages, using a stack instead of recursion speeds up your app tremendously."

yes i do understand. it is just less error prone to let the computer do the stack keeping for you. also, a good compiler will have a good optimizer which will make handcrafted optimization look slow (unfortunately, apart from tail recursion, recursion is not the favorite playground of optimizers).

but, as always, the solution used to solve the problem will depend greatly on the domain you are coding for. you will not code the same algorithm for an embedded system, a large-scale data-center or a desktop application. you will definitely have to make tradeoffs between resource usage, reliability, and ease of programming.

**Randolpho** Feb '07

Perhaps I'm failing my Sense Motive roll, but I'm truly saddened by the number of people who not only posted the solution, but posted the *wrong* solution.

Go back and count the "solutions" that failed to print the values from 1 to 100. Go back and count the solutions that started a loop at zero instead of one.

So sad. I really hope that they were deliberately wrong to keep the script kiddies from getting a job. In fact, my sanity requires that I chose to believe that.

**FizzBuzz** Feb '07

```
int main() {
printf("Get back to work!\n");
return 0;
}
```

**LKM** Feb '07

rien: I guess something was lost in translation then 🙂

Sure, creating your own stack has its own sets of problems, and it's not suitable to replace all usages of recursion. It's often